SRI International

February 2007

BALANCING THE NEEDS OF PERSONALIZATION AND REASONING IN A USER-CENTRIC SCHEDULING ASSISTANT

Technical Note 561

Prepared by

Dr. Pauline Berry, Senior Computer Scientist
Dr. Melinda Gervasio, Senior Computer Scientist
Dr. Bart Peintner, Computer Scientist
Dr. Neil Yorke-Smith, Computer Scientist

Artificial Intelligence Center
SRI International

APPROVED FOR PUBLIC RELEASE, DISTRIBUTION UNLIMITED

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE **FEB 2007**	2. REPORT TYPE	3. DATES COVERED **00-00-2007 to 00-00-2007**

4. TITLE AND SUBTITLE
Balancing the Needs of Personalization and Reasoning in a User-Centric Scheduling Assistant

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
SRI International,333 Ravenswood Avenue,Menlo Park,CA,94025-3493

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSOR/MONITOR’S ACRONYM(S)

11. SPONSOR/MONITOR’S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
http://www.ai.sri.com/pubs/files/1524.pdf

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**31**	

Balancing the Needs of Personalization and Reasoning in a User-Centric Scheduling Assistant

Pauline M. Berry Melinda Gervasio Bart Peintner
Neil Yorke-Smith*

Artificial Intelligence Center, SRI International
Menlo Park, CA 94025 USA
{berry,gervasio,peintner,nysmith}@AI.SRI.COM

Technical Note No. 561
February 2007

We describe the interaction of three aspects core to a personalized scheduling task. First, we develop a preference model designed to capture user preferences for the task of scheduling a meeting request between multiple people, and a methodology for preference elicitation to initially populate this model. Second, we explain a natural-language-based elicitation of the meeting request details and constraints, and outline the solving of the resulting constrained scheduling problem (with preferences). Third, we describe the display of solutions to the scheduling problem to the user, as candidate scheduling options with explanations, and detail unobtrusive learning of revisions to the preference model from the user's choices among the candidates. We describe the user studies that informed our design choices, and assess the resulting system in terms of the quality of scheduling options presented, according to the user. The scheduling task enabled by the integration of these aspects has been implemented within a deployed application.

*Contact author

1 INTRODUCTION

All too often, arranging meetings in an office environment turns into a tedious process. One common method that people employ is a series of emails to propose, reject, counter-propose, and eventually agree on a time. The effort consumed in such practices motivates the opportunity for automated assistance in scheduling.

Although a number of fully- or semi-automated scheduling systems have been developed, they have suffered from low adoption rates for two main reasons [14, 32, 40]: they fail to account for the personal nature of scheduling, or they demand too much control of an important aspect of an individual's working world.

The personal nature of scheduling is most directly seen in situations where a user's meeting request cannot be fully satisfied. For example, suppose that Alice requests a 90 minute meeting with Bob and Chris next Tuesday afternoon, but no time is available to all during that period. Some users may prefer the option of a shortened 60 minute meeting, while others (like Alice) may prefer the full 90 minute meeting Tuesday morning. Most users would like to be presented with both options (more generally, with all relevant options, of which there can be many in such *over-constrained* situations), but such that the options they most prefer have priority in the presentation. It is precisely in these difficult, over-constrained situations, in which there may be many competing factors and many possible *relaxations* of the meeting request, where scheduling assistance is most useful. But to be of real value here, an automated scheduling assistant must actively account for a user's scheduling preferences.

This report describes our work on modelling, eliciting, learning, and reasoning about the scheduling preferences of an individual. The approach we take is to combine initial, lightweight elicitation of scheduling preferences with unobtrusive, online refinement of them. This personalized preference model informs a constraint-based scheduling engine that computes and presents options in response to a meeting request. The user may select one of these options, request further options, or refine her meeting request. By initially eliciting the user's preferences, the system can offer reasonable scheduling options from first use; by refining its knowledge, the system can become progressively more capable and trustworthy over time [20].

While prior research has looked at one or more of these aspects — modelling and eliciting, learning, and reasoning — the resulting systems have rarely sought to encompass all three. For example, representing and learning user preferences but not performing sophisticated reasoning to offer scheduling options [25], or representing preferences and performing constraint reasoning but not updating the preferences by learning [13].

The key contribution of this report is a description and analysis of a preference model that balances the elicitation, learning, and constraint reasoning trade-offs within a semi-automated scheduling system. First, for elicitation, more expressive models better capture the nuances of user preferences, but require more effort to specify. Second, for learning, expressive models require significant training (and tend to overfit), while inexpressive models cannot distinguish between candidate schedules that are distinguished by the user. Third, for automated searching for and ranking of candidate schedules (hereafter, constraint reasoning), it is more difficult to define and reason over complex objective functions, preferences, and constraints.

Based on our preference model, the end-to-end semi-automated scheduling assistance enabled by the integration of these three aspects has been implemented within a deployed system called

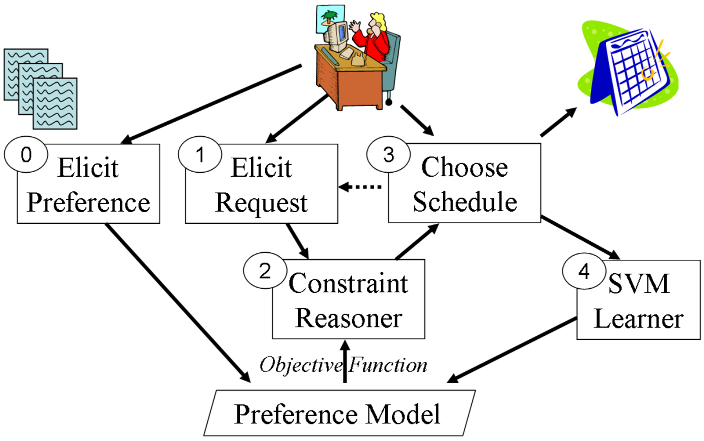


Figure 1: Use of the preference model in PTIME.

PTIME [2]. PTIME, a component of a larger cognitive assistant named *CALO* [29], manages the calendar and the scheduling requests of a CALO user in a mixed-initiative manner. Figure 1 shows the process. PTIME elicits scheduling preferences (step 0); elicits a meeting request (step 1); computes candidate schedules (possibly relaxations) in response to the request, by means of a Constraint Reasoner module, and displays a subset of the candidates to the user as options (step 2); and accepts the user's choice of the desired schedule option (step 3). Based on which option the user chooses for each request among the presented options, the learning module in PTIME updates the parameters of the preference model instance (step 4) [9]. The updated model becomes the basis of reasoning over candidate schedules for the next scheduling request.

The dashed arrow from step 3 back to step 2 indicates the user's ability to reject all the presented schedule options and revise her scheduling request. She might do so because she finds none of the options satisfactory, or because seeing the options has stimulated her to refine her request or explore an alternative.

At present, the PTIME user organizing the meeting decides which meeting option to select, taking into consideration others participants' generic scheduling preferences. In the future, she will be able to also take others' meeting-specific preferences into consideration. The selected meeting option is presented to invitees for inclusion or otherwise in their calendars. In the simplest form of negotiation supported by PTIME, the other participants besides the organizer may simply accept the meeting request or not. The system is being developed to support more elaborate forms of negotiation.

We begin by describing our first user studies and the requirements on the preference model that we derived from them. We then describe our model, based on Multi-Attribute Utility Theory (MAUT). [19]. Next, we describe the interfaces and design choices for eliciting both an initial instance of the model (*preference elicitation*), and the specification of and constraints on meeting requests (*problem elicitation*). The following sections describe the presentation of schedule options and the learning based on user choices, and then report a set of studies and experiments we conducted to validate (or otherwise) the three aspects of model and elicitation, learning, and reasoning, and the overall user experience of the PTIME system.

2 DEVELOPMENT OF A BALANCED PREFERENCE MODEL

An earlier model of user scheduling preferences in the PTIME system, reported in [9], sought to capture temporal preferences (day and time of events) for under-constrained meeting requests. It was composed as a weighted linear sum of features such as meeting start and end times. Building a constraint problem representation and reasoning over it, to find preferred schedule options according to the model, was straightforward. Refining a model instance, likewise, proved effective using *support vector machine* learning techniques [17]. Despite these positive aspects, the model could not capture non-temporal preferences, such as for meeting participants, and its expressiveness was too limited to capture how temporal and non-temporal criteria interact. Compared to the under-constrained case, these two aspects have a significant role for over-constrained meeting requests.

2.1 User Studies on Scheduling Habits

To gain insight into how individuals arrange meetings, and the factors that influence their decision-making regarding how to propose and respond to meeting requests, we conducted an informal user study. The study consisted of two parts: an in-situ diarying by the subjects, and a post-hoc evaluation of the diary combined with a semi-structured interview. We did not expect substantive quantitative data to result from the diaries; rather, our aim in the study was to understand the qualitative factors influencing the user's scheduling practice: both their individual nature and their mutual interaction.

Eleven subjects took part in the study, all members of our organization. Their roles included software engineer, engineering manager, researcher, program manager, and administrative staff. We asked the subjects to maintain a diary over one week's worth of scheduling activities, keeping track of how meetings were scheduled, what decisions were made and why they were made. We targeted some subjects with crowded, constrained schedules. Two of the subjects were unable to perform the diarying, either due to excessive busyness (despite the low overhead of the exercise), or because their schedules are managed by their administrators. We nonetheless interviewed these subjects regarding their scheduling preferences. In one case, we asked an administrator to perform the diarying and interviewed her with respect to scheduling for the individual she assists. The fivefold foci of the study were *event characteristics* (e.g., one-on-one vs. group meetings), *scheduling processes* (e.g., iterated refinement of a time), *decision factors* (e.g., relationship to meeting host), *preferences*, and *scheduling needs*.

Meetings/events We found that the subjects perceive three dimensions to the meetings on their calendars: (1) *one-on-one* vs. *group* (one-on-one meetings are often walk-in/impromptu while group meetings are scheduled in advance); (2) *mandatory* vs. *optional attendance* (the subjects often place optional events (e.g., seminars) on their calendar, with the understanding that they may be deleted/overlapped); and (3) *fixed* vs. *floating* (a sample floating event is "exercise at least twice a week").

Process We found that the subjects schedule events in a variety of ways, including: (1) *walk-in* (most often used for one-on-one and for critical meetings); (2) *constraint satisfaction* (used for group meetings: e.g., host suggests a time window and asks for everyone's constraints; if there is a viable time, the meeting is set; otherwise, the process repeats); (3) *iterated refinement* (used for one-on-one meetings: e.g., A suggests tomorrow, B says afternoon, A asks "2pm?", B responds "how about 3pm?", A says okay); (4) *add-in* (used for meeting announcements and recurrent meetings); and (5) soft scheduling, i.e.not placed in the calendar (often used for tasks/todos and other floating events).

Factors We found that the subjects take into consideration a handful of factors that are a subset of a broadly common list, when scheduling a meeting and when deciding whether to accept a meeting request. We group the factors into seven categories: (1) importance (of the meeting; its importance to you, your importance to the meeting); (2) urgency/criticality; (3) interest; (4) relationships (host, relationship to host; other participants, relationship to other participants [40]); (5) perturbation (effect on other meetings); (6) stability (often characterized as "number of times the meeting has been rescheduled"); and (7) preferences (of the individual and of others). The relative importance of the features varied considerably between subjects.

Preferences We found that the subjects explicitly indicated preferences for or against some specific features. We group them into four categories: (1) general meeting-specific preferences (e.g., time of day); (2) general calendar-wide preferences (e.g., fragmentation, density, number of meetings per day); (3) preferences over how to relax meeting request constraints (e.g., proximity to specified time, proximity to specified duration, attendance of high-priority participants); and (4) preferences over how to relax calendar-wide constraints (e.g., no overlaps, no Friday p.m. meetings because of childcare situations, no late meetings on carpool days).

Needs We concluded the interviews by asking "What do you most want out of a scheduling assistant?" The most common responses were: (1) coordinating meetings between a group of busy people; (2) intelligent reminders; (3) transparency into assisted scheduling decisions (e.g., explanations of conflicts and learned preferences); and (4) greater control over scheduling processes (e.g., accepting requests, separation of participants' availability and preferences).

Among other studies of user scheduling habits and (semi-)automated calendaring tools, Palen [32], for instance, examined the use in situ of group calendaring software at Sun Microsystems. Several researchers report that people are reluctant to invest in accurately and fully informing a scheduling system of their preferences (to the extent that they can articulate them) unless either (1) the process is not burdensome and they are persuaded of the benefit; or (2) they are mandated to do so [14, 40]. Other studies support our finding that even when people are confident in the behavior and the decisions of a (semi-)automated system, they seek transparency into its reasoning [1, 32].

Our own study and prior work also demonstrates that evaluation of scheduling options is contingent on multiple criteria and their interaction [20, 3]. Accounting for the relative importance of these features is crucial if we are to offer the user desirable relaxations for over-constrained requests. Whereas [13, 9, 26] and others assume no interaction between the criteria in their preference model — which simplifies the learning and solving aspects (to the extent that earlier work accounting for preferences makes use of both learning and reasoning) — this choice limits the expressiveness of the model.

A set of requirements for a preference model results from these various ethnographic investigations. First, the model must be populated from intuitive (and thus likely qualitative) statements expressed in terms of concepts that the user is familiar with in the domain, i.e., events, people, and calendars. Second, the expressiveness of the model must be such that it can capture enough of user scheduling preferences to enable the personalized scheduling task. Third, the preference model must be explainable to the user, again in terms of familiar, domain-relevant concepts [25]. Fourth, the preference model must be able to express multiple criteria that factor into the user's decisions, and the interaction between the criteria, such as between the meeting duration and the temporal preferences of other participants. Finally, as we have argued, the model must be tractable for reasoning and learning, if a practical scheduling assistant is to be built.

2.2 Choquet Integral Model of Scheduling Preferences

The preferences literature is rich with qualitative and quantitative models of varying expressive power and computational tractability; for surveys, we refer to [31, 10]. The reasoning and learning methods we wish to bring to bear on scheduling problems are quantitative, and the features of the preferences are not independent. Suitable models can be based on, for instance, Generalized Additive Utility (GAI) [7] or Multi-Attribute Utility Theory (MAUT) [19].

To balance the competing aspects of expressiveness and elicitation, learning, and reasoning, and to meet the above requirements, we chose to adopt the global utility function approach of MAUT. By providing a single function by which the system can rate alternative schedules, a MAUT approach is in principle amenable to the schedule evaluation and preference learning components. The challenge of adopting any approach is to build a model suited to the scheduling domain, and to adapt reasoning and learning algorithms to it.

A MAUT model is specified by a set of n criteria, a set $u_1, \ldots, u_n$ of (partial) utility functions that make the criteria commensurate, and an *aggregation function* F. An instance of the model, i.e., a capturing of the preferences of an individual, is specified by the coefficients of the aggregation function. For example, writing $z_j = u_j(x_j)$, where x_j is a value for criterion j, then $F(z_1, \ldots, z_n) = \sum_i a_i z_i$ for some set of coefficients a_k, corresponds to aggregation by a linear weighted sum.

A weighted sum cannot express interaction between criteria; it assumes that all criteria are preferentially independent [19]. An aggregation function that avoids this assumption and that satisfies certain desirable properties is the *Choquet integral* [12, 21]. The Choquet integral subsumes a weighted sum, and is able to express multi-criteria trade-offs such as Pareto optimal decisions that a weighted sum cannot represent. We now explain how we define a scheduling preference model based on this representation.

From our first user study reported above, augmented by features suggested in prior work in the literature (e.g., [20, 25, 3]), we identified seven criteria consistent across different users: (1) scheduling windows for the requested meetings; (2) durations of meetings; (3) overlaps, ordering constraints, and conflicts between requested and existing meetings; (4) locations of meetings; (5) participants in meetings; (6) time or duration changes for existing meetings; and (7) preferences of others participating in new meetings or rescheduled existing meetings [8].

We deliberately chose criteria expressed in terms of concepts familiar to the user in the domain to facilitate elicitation of instances of the preference model and explanation of learned preferences. Below we describe the formulation of the commensurate utility functions u_i. Other criteria, including the stability of a candidate schedule (how stable it is with respect to new meetings and meetings that run long) and how much the candidate schedule perturbs the existing schedule, would add richness to the preference model. However, as we will describe, we found that the richer model would not be amenable to constraint solving.

With the criteria specified, we next define the Choquet integral aggregation function simplified to this context.

Definition 1. *Let $z_i = u_i(x_i)$, let N be the set of criteria, and let $I \subseteq N$ be a subset of the criteria.. The* Choquet integral *is*

$$F(z_1, \ldots, z_n) = \sum_{I \subseteq N} a_I \bigwedge_{j \in I} z_j \tag{1}$$

where a_I is a coefficient representing the degree and type of interaction of the criteria in I, and where $\wedge$ denotes conjunction. □

Over n criteria, specification of the general Choquet integral requires 2^n coefficients. A *k-additive* or *k-order* Choquet integral considers only the interactions of criteria sets with k or fewer criteria. It trades expressiveness of the model for its easier specification. Practical applications indicate that the 2-order case is usually sufficient [24, 22]. Only $n + \binom{n}{2} = \frac{n(n+1)}{2}$ coefficients are required to specify the integral. In this case, (1) can be written as

$$F(z_1, \ldots, z_n) = \sum_{i \in N} a_i z_i + \sum_{\{i,j\} \subseteq N} a_{ij}(z_i \wedge z_j) \tag{2}$$

where the coefficients a_i, $i \in N$, and a_{ij}, $\{i, j\} \subseteq N$, fully specify the model. The conjunction operator $\wedge$ can be specialized to minimum. Details of the derivation are given in [24].

In the 2-order case, the coefficient $a_i \in [0, 1]$ describes the relative importance of criterion i (a greater value indicates greater relative importance), while $a_{ij} \in [-1, 1]$ describes the interaction between criteria i and j. $a_{ij} > 0$ indicates that i and j are complementary criteria, while $a_{ij} < 0$ indicates that they are substitutive; $a_{ij} = 0$ indicates no correlation between the two.

We make the hypothesis that a 2-order model trades off expressiveness (being unable to express interaction effects among three or more criteria) with ease of model specification and explanation in a way suitable for (1) a constraint-based representation of the scheduling problem, (2) reasoning over this problem representation, and (3) learning revisions to the model. With $n = 7$ criteria, an instance of the model is specified by $\frac{1}{2}7(7 + 1) = 28$ coefficients. Note, however, that the reasoning and learning described in the sequel do not depend on a 2-order model, other than their complexity increases as the number of Choquet coefficients increases.

3 PREFERENCE AND PROBLEM ELICITATION

Both preference and problem elicitation share the common challenges of eliciting preference information from a user. On one hand, information that the user does know and can express she may not be willing to input to the system, especially if the elicitation process is burdensome. On the other hand, the user may not fully know the information or be able to express it (at least via the elicitation mechanism offered). Moreover, the very process of elicitation itself is known to reshape — and at worst bias — the information provided; preferences can even be created by the elicitation process [33, 41].

Preference elicitation One approach to elicitation of general scheduling preferences is to derive rankings of schedule options by showing the user specific examples as part of an elicitation phase. *groupTime* [3] is a scheduling system that takes such an example-driven approach. Viappiani et al. [41] show that presenting users with carefully chosen examples can help stimulate their preferences, a technique known as *example critiquing*.

An alternative approach to elicitation has the user enter the parameters of the preference model directly. The scheduling system of Hayes et al. [13] takes such a model-driven approach. The advantage of this approach is that the elicitation period can be shorter, because the information elicited has greater entropy. The significant disadvantage is that users must comprehend the model, rather than examples that are possibly easier to comprehend.

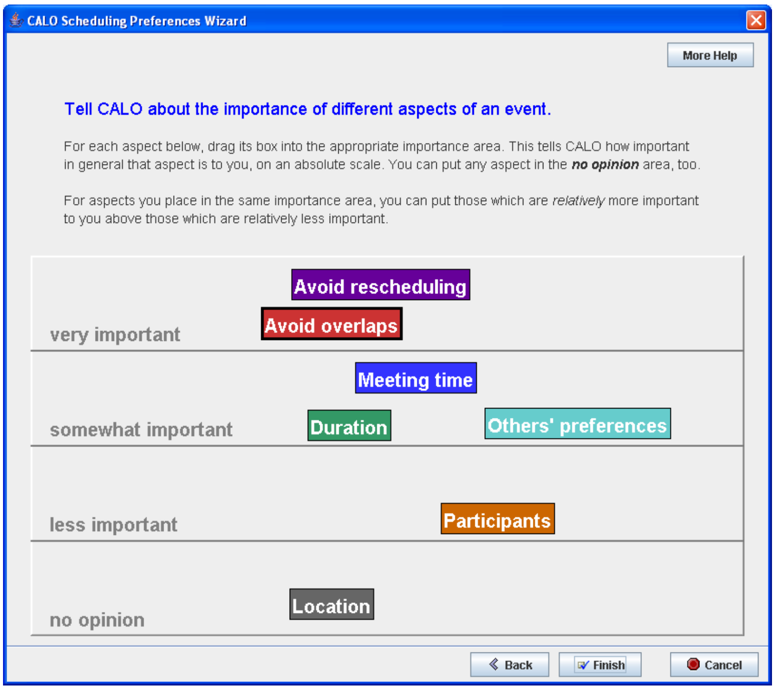
The coefficients of a Choquet integral, the goal of our elicitation, can be derived from statements over examples (*"I rate this example more highly than that"*) or over the model (*"overlap is more important for me than duration"*) [24].

We chose to explore a form of model-driven preference specification for three reasons. First, we considered that the user's intuition of the scheduling domain reduces the cognitive effort to understand the model, in contrast to the need to digest an artificial example before giving meaningful feedback based upon it. This is especially true for over-constrained examples, where a perception of the calendars and preferences of involved participants is necessary to make an informed judgment over scheduling options. Second, the use of online learning in PTIME updates the model from examples (real-life examples for which the user already has in mind the meeting request, and participants' calendars and preferences, to some degree). Third, we considered it advantageous to enable the user to employ PTIME without an extended elicitation phase, by developing a one-shot, model-driven elicitation.

Hence, our approach to preference elicitation combines an interactive, visual interface and a series of simple elicitation *invitations*. Each invitation is presented as a panel, such as in Figure 2, that asks the user to provide some statements of her general scheduling preferences. The panels are presented in a 'wizard' interface that allows the user to freely step forward and backward through them, and to ignore any invitation (i.e., provide no information for it).

The first panel invites the user to describe her general temporal preferences over a week, by painting them onto a calendar view 'canvas'. This direct elicitation of temporal preferences follows prior calendaring agents, such as [13] and contemporary work, such as [3].

From the information entered by the user, we infer qualitative preference statements on and between the criteria, and compile these statements into quantitative coefficients in the Choquet representation. This compilation is performed by solving a linear program (LP); for the details



Tell CALO about the importance of different aspects of an event.

For each aspect below, drag its box into the appropriate importance area. This tells CALO how important in general that aspect is to you, on an absolute scale. You can put any aspect in the ***no opinion*** area, too.

For aspects you place in the same importance area, you can put those which are *relatively* more important to you above those which are relatively less important.

Figure 2: A single panel of the criteria preference elicitation wizard. Users are asked to position each criterion to reflect its importance relative to both other criteria and qualitative importance labels.

we refer to [24]. The preference statements we infer encompass information regarding both the relative importances of the criteria (i.e., the a_i Choquet coefficients), and the trade-offs between criteria pairs (i.e., the a_{ij} coefficients).

In the panel shown in Figure 2, the classification of the criteria into the four buckets speaks to the first aspect (a_i), and the pairwise relative positions of the criteria speak to the second aspect (a_{ij}). For instance, from the position of the overlap criterion in the 'very important' bucket, the system will infer a value for the corresponding a_3 coefficient close to 1. From the relative positions of this criterion and the duration criterion, the system will infer a value for the a_{23} criterion less than zero.

This two-stage approach of interactive elicitation followed by compilation is designed to blend intuitive, lightweight elicitation for the user, in qualitative, domain-dependent terms familiar to her, with the eventual derivation of quantitative coefficients required by our model. Moreover, the process is robust to the amount of information, falling back to an uninformative, default instantiation of the model in the case of no information.[1]

It is well known (e.g., [19]) that people do not act rationally and often do not have consistent desires. Thus, the preference statements made may result in conflicts. In these cases, we iteratively relax or remove conflicting statements until the resulting LP is consistent.[2]

[1] Equal importance ($a_i = 1/n$) to each criterion; no interaction ($a_{ij} = 0$) between each pair of criteria.

[2] We judged that any benefit from including the user in this process would be outweighed by the burden of possible added confusion. Relaxation of inconsistent Choquet models is a research topic beyond our scope here [23].

Problem elicitation Like preference elicitation, our approach to problem elicitation is based on an interactive interface. Common office calendaring tools, such as Microsoft Outlook, obtain meeting details by allowing the user to select a block of time on a calendar interface, and then filling in details in a form. Recent tools, such as Google Calendar [11], allow the user to specify the meeting details by entering natural language (NL) sentences. In both cases, such tools are not seeking to elicit information in order to formulate and solve a problem of presenting schedule options, but simply to fix a meeting in the user's calendar.

The information for our scheduling task can be both broader (*"An afternoon next week"*) and more specific (*"Bob is an optional participant"*) than that required by calendaring tools. We are seeking to elicit not the details of the meeting itself, but details of the meeting request — a potentially rich set of soft constraints that will be used to formulate a constrained scheduling problem instance. Since the constraints are relaxable, problem elicitation can be seen as elicitation of problem-specific (in contrast to generic scheduling) preferences.

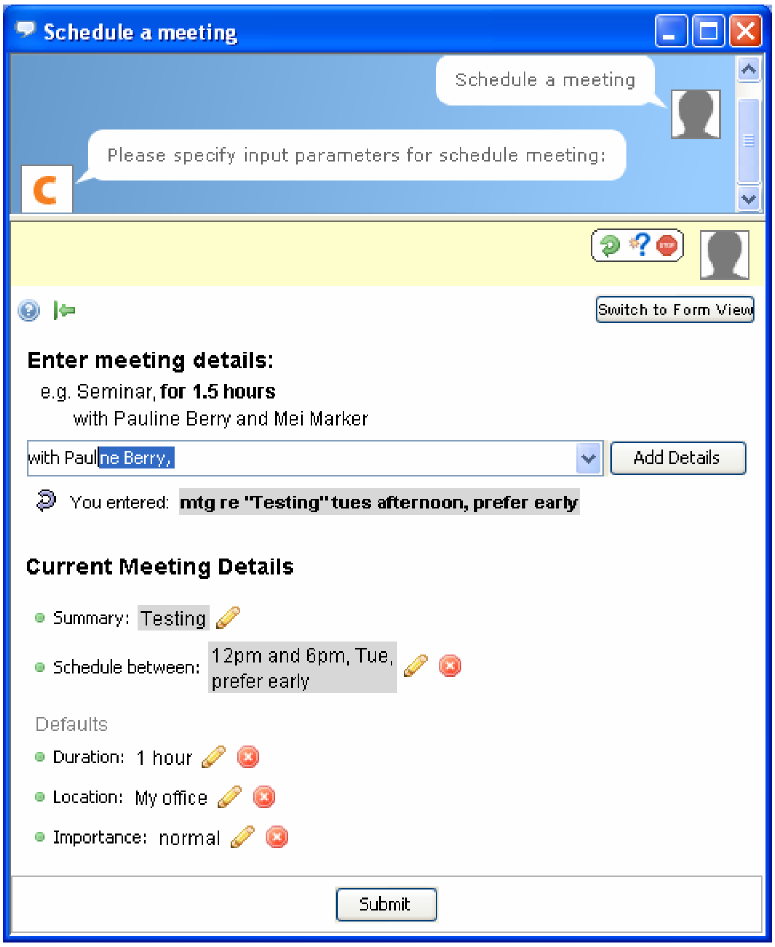
Thus, the deficiencies of common approaches are magnified. On one hand, form-based elicitation varies from restrictive, since the user is limited to the fields in the form, to intimidating, if the form contains enough fields to allow expression of rich constraints. Further, users are observed to feel they must fill in a field simply because of its presence [33]. On the other hand, unrestricted NL can leave the user unsure of what to enter, especially once the illusion that the system really understands everything entered is (inevitably) broken. Moreover, while forms place restrictions or requirements on the user, NL provides little guidance: for example, that the user can specify optional participants.

Figure 3 shows our problem elicitation interface. In the top section of the window we see a system-user dialogue, following DiamondHelp [37]. In the bottom section of the window we see dynamic, context-specific content — here the meeting request interface. Based around an NL input mechanism (centre), the system summarizes the information the user has entered already (bottom), and stimulates her with 'example' lines (top). The design choices and rationale behind them are discussed in [2].

Because of the NL interface, the user can readily specify *soft* constraints (i.e., constraints that may be satisfied only to a degree, in contrast to *hard* constraints that must be satisfied exactly), such as preferred times (e.g., *"prefer early"*, or *"prefer 3pm"*), and optional and preferred locations and participants. The user can also flexibly and succinctly specify time windows (e.g., *"tues afternoon"*).

Transparency and predictability are enhanced by the system reporting what it understood: note the "You entered:" message and the highlighted fields, which provide clarification of what the system understood. This can be contrasted with the more heavyweight NL clarifications in RhaiCAL [5]. Our clarification minimizes the cost of poor guesses by the system (another principle of Horvitz) by making explicit what it understands, at a glance. Further, auto-completion removes the burden of typing in (and possibly misspelling) locations, participant names, and so on. This lessens another characteristic problem of a natural language modality.

The NL interface provides a mechanism for specifying preferences that is designed to be superior to a form-based direct manipulation input interface. Nonetheless, since easy access to direct manipulation is among the principles for mixed-initiative user interfaces [16], and recognizing that some users find a form-based interface more familiar, we provide an option to "Switch to Form View" that allows direct input of a subset of the possible constraints.



Schedule a meeting

Please specify input parameters for schedule meeting:

Enter meeting details:

e.g. Seminar, **for 1.5 hours**
with Pauline Berry and Mei Marker

with Pauline Berry,

You entered: **mtg re "Testing" tues afternoon, prefer early**

Current Meeting Details

- Summary: Testing
- Schedule between: 12pm and 6pm, Tue, prefer early

Defaults

- Duration: 1 hour
- Location: My office
- Importance: normal

Figure 3: Meeting request (problem) elicitation interface. The "You entered:" line displays the previously entered text, while the "Current Meeting Details" show how the system parsed the text.

Our interface is suited to specification of straightforward temporal preferences for the meeting, such as a combined preference for the morning over the afternoon, and later in the morning over earlier. It is less suited for specification of finely grained preferences, such as *"later between 10–11am, or as soon as possible after noon, but not between 11–11:30am"*. As the last sentence exemplifies, such highly detailed preferences are cumbersome to describe in text. Rather, a dedicated direct-manipulation interface for such preference would be preferable [6]. We have not implemented this complementary kind of interface, because our first user studies indicated that meetings are rarely requested with such detailed preferences. We think it better for the user to retain these preferences and use them to guide among relaxations of the meeting request, if warranted, when the schedule options are presented by the system.

Fundamentally, the scheduling domain scopes the statements the user wishes to input so that restricted NL is practicable, and directs the statements, so that it is intuitive. Altogether, the assisted NL interface was favourably received by users in the restricted setting of our calendaring domain, relative to the direct manipulation interface. Our user studies are reported below in Section 6.

In the background, while the user is specifying the meeting request, the system retrieves the schedule of the user and the free/busy times of all specified rooms and participants. Lightweight constraint reasoning is performed after each change to detect conflicts and display them with suggestions for removing them. The calendaring domains allows specialized partial satisfaction and explanation mechanisms [18].

We provide an option for the user to display her current calendar. The calendar display includes options to overlay the day/time preferences and free/busy availability of participants requested in the meeting so far. Combined, the conflict display and repair suggestions, and the calendar and preference display, inform the user as she composes her meeting request.

4 CONSTRAINT REASONING

The aim of constraint reasoning is to generate candidate options in response to a scheduling problem instance that results from problem elicitation (recall Figure 1). This reasoning must account for the preference model of the user, the constraints and preferences in the scheduling problem, and both the current schedule and day/time preferences for all involved participants.

To find desirable solutions according to our chosen preference model, the PTIME Constraint Reasoner must search for candidate schedules using the Choquet integral as the objective function. Thus, given our preference model and the state of the art in constraint-based temporal reasoning, we have two questions to address: (1) how to represent the hard and soft temporal constraints in a form that translates into the criteria we have chosen; and (2) how to extend the temporal reasoning beyond the simple objective functions found in the literature to the more complicated Choquet integral.

Both questions arise from the core issue of this work: balancing the need for an expressive preference model with difficulty of learning and reasoning about it. In previous quantitative constraint-based temporal optimization work, objective functions were based on simple aggregations of *local preference values* — values indicating how well each constraint is satisfied (e.g., [35]). These objectives do not easily extend to encompass a preference model based on a Choquet integral. First, our model is based on abstract (domain-relevant) criteria, not individual constraints. Second, the model contains interactions between criteria, something not expressible in existing frameworks, even for the case in which each constraint is considered a distinct criterion.

Our reasoning approach has two facets that correspond to the two questions above. First, we map each constraint to a subset of the criteria, based on the origin of each constraint. For example, a soft constraint expressing allowed durations maps to the duration criterion, and a constraint expressing that a person cannot attend two meetings at once maps to the overlap criterion. The mapping is one to many: a constraint can map to multiple criteria.[3]

To give a sense of how we represent the soft constraints in the meeting problem, Figure 4 depicts a constraint, and preference functions, for an overlap constraint between an existing meeting (E) and a new meeting (M) that both involve a common participant. In a hard (i.e., must be satisfied exactly) version of this constraint, either the start of the existing meeting must occur after the new meeting or the start of the new meeting after the existing meeting. In the depicted soft version, the preference functions allow for the events to overlap but specify that a smaller is preferred.

Formally, we model the constraints as a *Disjunctive Temporal Problem with Preferences* (DTPP) [35]. Adding the mapping from constraints to criteria transforms the DTPP into a *Multi-Criteria DTPP* (MC-DTPP) [27]. To optimally solve an MC-DTPP, a solution must be found that maximizes the value of the Choquet integral equation (2). To optimize over the full integral (which, recall, is a sum of as many as 2^n terms) would be extremely challenging. Every search point would require significant time to calculate the objective value and, more important, it would be difficult to develop effective heuristics or pruning strategies.

[3] An alternate and equivalent view is that a single criterion maps to many constraints. Viewed either way, there is similarity with GAI preference models [7], in which overlapping subsets of attributes compose each factor.

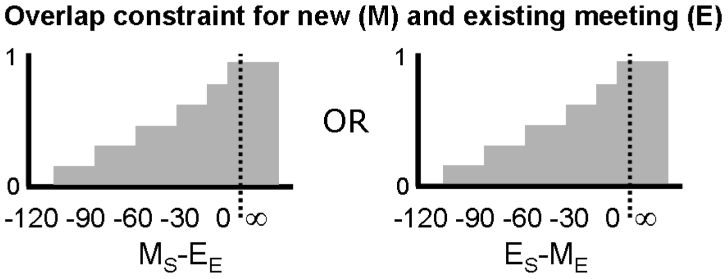


Figure 4: A DTPP constraint that represents the soft overlap between an existing meeting E and a new meeting M involving the same person.

This difficulty highlights the trade-off between model expressiveness and tractable reasoning. It motivates our choice of the 2-order Choquet integral, which captures at least one level of criteria interaction, but can be represented using the sum of only $\frac{1}{2}n(n+1)$ terms. It also guides our choice of criteria: criteria such as stability and perturbation mentioned earlier are functions of an entire schedule, and thus do not easily incorporate with the standard DTPP scheme of aggregating over *local* preference values.

The second facet of our reasoning is thus an effective algorithm for solving an MC-DTPP to obtain the scheduling options. The algorithm augments a leading branch-and-bound DTPP solver [28] with special bounding logic and additional heuristics. We leave the details to [27].

5 SCHEDULE PRESENTATION AND LEARNING

From the many solutions generated by the Constraint Reasoner we want to present a subset — the candidate scheduling options — to the user. On one hand, the system must not overwhelm the user with too many options, but on the other hand, it must present enough options so that the user can select one that is acceptable. In general, we want to present the most desirable solutions first (according to the elicited preference model). However, the model will generally be an imperfect approximation of the user's "true" preferences, so we also want to present additional solutions that are not necessarily rated highly but that are qualitatively different from those thought most preferred according to the current model instantiation. This approach presents desirable solutions and at the same time also enables the user to explore the solution space. With each option that relaxes the meeting request, we provide an explanation in the form of a simplified, natural language summary of the relaxed constraints.

While the elicited preferences provide a starting point for presenting solutions tailored to the user, an effective scheduling assistant should refine its preference model over time. Thus, we employ machine learning techniques. It would be disruptive to the user experience to interrupt with learning questions; therefore, in a deployed setting, learning must happen online using only feedback obtained through a user's natural interaction with the system — i.e., through the user's selections from among the candidates presented; these are the training examples for the learner.

Recall that in our earlier work, the PTIME Preference Learner acquired user preferences over temporal schedule features (i.e., day/time preferences) using *support vector machine* (SVM) learning techniques for ranking [17]. The learner acquired the weights of a linear schedule evaluation function that could be used to rank candidate schedules [9].[4]

Our shift to multi-criteria schedule evaluation fundamentally changes the learning task. While the task remains to learn a schedule evaluation function, the features of the function are no longer boolean features representing temporal properties of a schedule, but instead are real-valued features representing the degree of satisfaction of higher-level criteria u_i, each of which is itself a function of lower-level schedule features, as described earlier. Moreover, the function being learned is a 2-order Choquet integral, rather than a linear weighted sum. This adds the constraints that the coefficients obey $a_i \in [0, 1]$ and $a_{ij} \in [-1, 1]$, as mandated by the Choquet model [21, 24].

Observe that the 2-order Choquet integral can be viewed as a linear weighted sum of a *new* set of features (criteria) comprising affine combinations of the importance coefficients a_i and interaction coefficients a_{ij}. Based on this linearization, we can apply the same SVM learning techniques as earlier to learn the coefficients (weights) for the resulting transformed function, subject to the constraints on the values of the coefficients. Because the SVM learning problem is essentially a quadratic optimization problem, to ensure that the coefficients learned are valid Choquet coefficients, we could augment the quadratic programming problem with additional constraints. However, simple rescaling of the learned weights into the range $[-1, 1]$ is sufficient for satisfying the constraint on the interaction coefficients, and we found that the SVM learner naturally learns positive weights for the importance criteria just from the training exam-

[4] In that work, our initial focus was on learning preferences in under-constrained situations, although the same approach could have been used in principle to learn preferences in the more difficult over-constrained situations as well, by adding features to represent the degree of satisfaction of the different scheduling constraints.

ples themselves. Thus, in the end, no modification of the input to the SVM learning algorithm proved necessary.[5]

Since both the elicited and learned preferences have the same functional form, we use a simple weighting scheme to combine them into the refined schedule evaluation function

$$F'(Z) = \alpha \times A \cdot Z + (1 - \alpha) \times W \cdot Z \tag{3}$$

where A is the vector of coefficients (weights) for Z representing the elicited preferences and W are the learned weights for Z. By varying α we can vary the relative influence the initial and learned preferences have on the evaluation of a candidate schedule. By decaying α over time, we can dynamically modify this relative weighting to give more consideration to the learned weights as the learner views more training examples, i.e., as the user employs the system.

Our conversion of the Choquet model to a linear function over an expanded set of features ensures that we can continue to use a linear kernel. One advantage of linear kernels is that the resulting learned function is more easily interpreted. (An alternative to the feature expansion approach would be to change kernel.) However, the feature expansion does lose the connection between the importance criteria a_i and the interaction criteria a_{ij}: as far as the learner is concerned, it is learning relative weights for independent features. Thus, there is no guarantee that the resulting learned function will be a well-formed Choquet integral and, in fact, we cannot convert the learned weights back to valid Choquet coefficients. However, the MC-DTPP solving algorithm discussed in the previous section does not rely on a valid Choquet form but can accommodate a general MAUT objective function. Hence, provided that our learned function successfully captures user preferences, we can generate and present desirable scheduling options.

It is worth emphasizing that, after the transformation of the model, we work entirely with the linearized model. Both the constraint solving and the online learning work with the new representation. Thus the loss of connection between the a_i and a_{ij} does not hinder the computation over nor refinement of the model. However, it does present a greater challenge in explaining the current state of the model instance to the user, since techniques for explaining Choquet models are based on the original, untransformed coefficients. Providing a user-intelligible explanation of the current learned model is part of our ongoing work.

[5] In the few cases where the coefficients for the importance criteria are negative, they are very small and simply bringing them up to zero or a small positive number did not affect learning performance.

6 VALIDATION

To assess the value of our approach to the trade-off between representation, reasoning, and learning, we designed experiments to answer the following questions: (1) will the learning algorithm converge to a given preference model given consistent feedback; (2) how well do users' elicited preferences match their "true" preferences; (3) how well does the Preference Learner perform using feedback from real users on interesting scheduling instances; (4) does the Constraint Reasoner provide optimal solutions in an adequate amount of time; and (5) how well does the entire system perform in real-world situations?

6.1 Preliminary Evaluation of Learning Algorithm

To validate the learner and to help prepare for deployment, we ran a set of experiments using synthetic data — both randomly generated target functions (converted Choquet integral coefficients) as well as randomly generated schedules (feature vectors Z). Each experiment simulated an online learning situation where, at each iteration, the learned model ranked a set of candidate schedules, the target model selected the best schedule, and the learned model updated with this feedback in preparation for the next iteration. Based on the evaluation metrics discussed in the next sections, the results of these experiments confirm that the SVM learner can successfully learn the target functions in this setting, often within 10 to 20 scheduling episodes, each involving 25 candidates. As the number of candidates is reduced, however, the number of iterations required for convergence greatly increases. Figure 5 shows the rate of convergence for 5 and 25 candidates per iteration.

To test the effects of the quality of elicited (initial) preferences on performance, we varied how well the initial preferences matched the true target model and then analyzed the effects of different initial weights and decay strategies. The results suggest that instead of using a fixed weighting scheme between the elicited and the learned preferences, we should consider the quality of the elicited preferences in determining both the initial weight to give them and how aggressively to decay that weight over time. Further experiments, including those reported below, led us to choose an exponential decay.

6.2 Evaluation of Preference Model and Elicitation

To evaluate whether our preference model was able to capture the essential aspects of an individual's scheduling preferences, and whether our elicitation interface was adequate for users to initially express their preferences, we conducted a user study consisting of two parts: a session of preference elicitation with the interface described in the previous sections, and a series of simulated meeting requests, conducted as a paper-based exercise.

We asked 11 subjects — with roles including software engineer, researcher, program manager, and administrative staff — to perform a brief interaction with our system to elicit an instantiated model of their preferences. Some of these subjects had participated in our user habits study reported in Section 2.1; others had not. We then gave the subjects a hypothetical two-week calendar of a knowledge worker, containing a moderate number of meetings, events, and deadlines. We presented the subjects a series of meetings to organize. For each of the 11 meeting

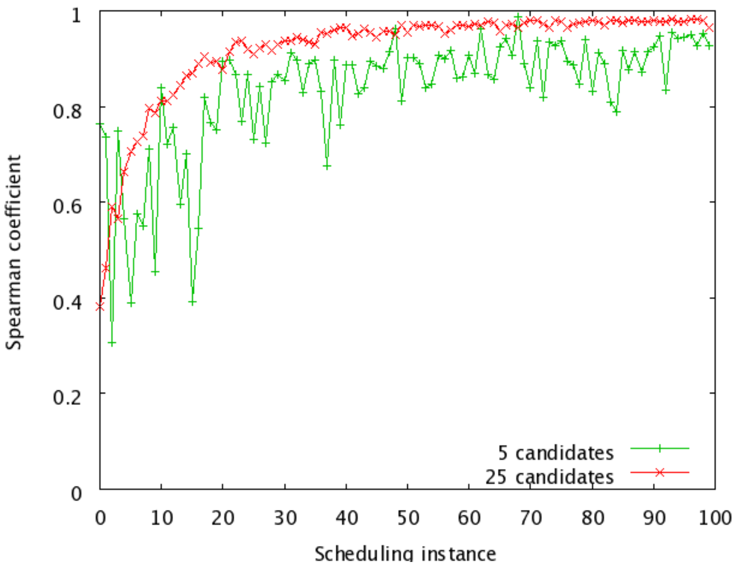


Figure 5: Spearman's correlation coefficient as the number of scheduling instances increases, for 5 and 25 candidates per instance.

requests, we asked each subject to rank 10 options, ties permitted. In an experimental oversight, each subject was presented with the requests in the same order. However, our later repeat of a subset of the experiment causes us to believe that the order of the meeting requests did not have significant impact on the options selected for each request by the subjects.

In contrast to the experiment in the previous section, the aim of this study was to investigate the extent to which elicited (initial) preferences correlated with *actual* ("true") preferences, which are unknown. We based a measure of the subjects' actual preference on the ranking of choices the subjects made in each presented meeting request. To quantify the degree of similarity between the elicited and the actual preferences, we used the Spearman's correlation coefficient ($\rho \in [-1, 1]$) [15] between the rankings provided by the subjects and the rankings induced by the elicited models. A ρ value of 1.0 indicates two identical rankings, while $\rho = -1.0$ indicates complete disagreement.

Overall, we observed a weak correlation between the actual and modelled preferences. The arithmetic mean of ρ across subjects and requests is 0.26, with a standard deviation of 0.32. The variance between subjects is substantial: the mean ρ value (across the subjects, averaged over the requests) varies from 0.08 to 0.42.

Figure 6 depicts the variance of ρ for each meeting request. The vertical lines show the range of ρ values over the subjects, and the boxes indicate one quartile from the mean. Observe that the weakest correlation is for Request 3, which is the only request for which the mean ρ is negative (-0.02). Request 3 ("All day travel to LA office") is a situation where factors outside those included in our preference model are significant, and thus correlation may be expected to be poor. By contrast, the strongest correlation is for Request 6 (0.46), which is a situation with delicate trade-offs between time and participants, the type of situation to which our model is specifically dedicated. These facts indicate that our chosen features work satisfactorily for

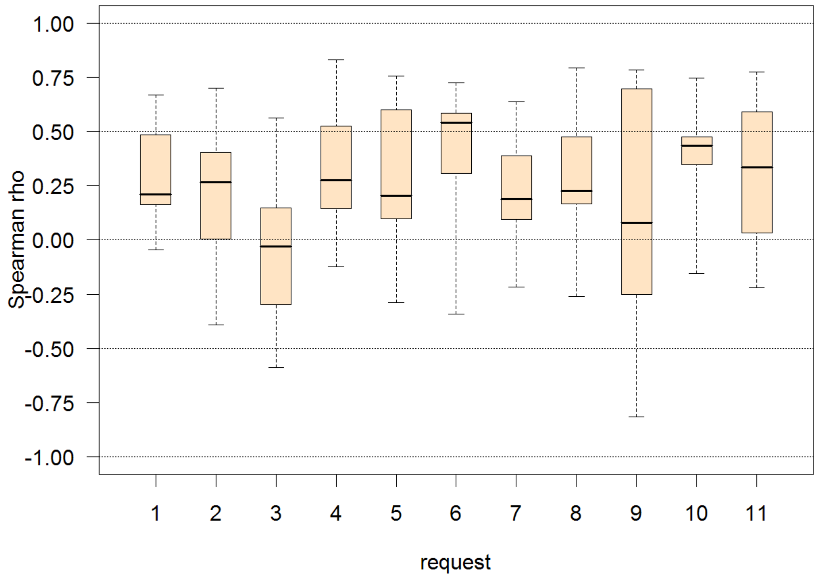


Figure 6: Spearman's correlation coefficient for each request and subject, grouped by request.

the common case. We could add features to address the poor correlation for Request 3, but the additional features may slow the rate of convergence for the learning and will make constraint solving more difficult. In free-form interviews after the study, subjects expressed satisfaction over the features included in the preference model.

The least variance over subjects is Request 1 (0.21 standard deviation), the most is Request 9 (0.48 standard deviation). Request 1 is a simple situation where most of the options are favorable; here, if the elicited model correlates well with the true preferences of one subject, it is likely to do so for all subjects. Request 9 is distinguished by conflicts among the preferences of other participants; here, the subjects assess the value of the options differently according to how they balance the importance of others' preferences.

To obtain a baseline for these Spearman ρ values, we compared each subject's elicited ranking to the ranking produced with two other means of instantiating the model: a default instantiation that corresponds to equal weights for all criteria, and a random (and possibly invalid) instantiation in which all Choquet coefficients are chosen randomly. This exposes the extent to which the correlation is due to the adequacy of the elicitation as opposed to the form of the model, or any other variable.

The results showed a mean Spearman value (across all subjects and requests) of 0.28 for the default and 0.27 for the random, both of which are slightly better than our elicited average; the standard deviation was 0.30 and 0.31, respectively (about the same as for elicited initialization). If we separate out the five subjects who had seen and used the interface before from the six who had not, we notice a substantial spread in the Spearman averages: the experienced users had an average of 0.33 (higher than the default), whereas the inexperienced users had an average of 0.19.

We might attribute this result to the elicitation interface, to the subjects' inability to comprehend their statements in terms of real-world scenarios, or possibly to the subjects' lack of understanding of their preferences. Informal discussions supported the latter hypothesis: several subjects reported that the process of ranking schedule options in the exercise helped them realize their own scheduling preferences, in particular over the trade-off of different criteria. As one subject stated, this indicates that individuals have a poor grasp of their actual preferences in this area; it suggests we should expect a substantive gap in correlation between elicited and actual preferences. We allowed two subjects to restate their preferences using our elicitation interface; both restatements resulted in a small increase in the Spearman correlation (by 0.02 and 0.06, respectively), which agrees with the results we found with the experienced users above.

To further examine the variance in the results, we asked five of the subjects to perform a second paper-based exercise. Unknown to these subjects, the second exercise was equivalent to the first, but with textual descriptions, request order, and schedule option order varied. Of these five subjects, the mean Spearman correlation decreased for four, and increased for the last subject. The variance is approximately unchanged for each subject. Only one of these users had a significant deviation (-0.18), although this user was unique given that he interpreted the exercise much differently than others (he did not rank any schedule that he deemed "unacceptable"). These results indicate that there is a moderate but not dominating perturbation factor in the study arising from the subjects' inconsistency in their scheduling decisions.

Overall, the small but positive correlation between the actual and elicited preferences can be explained by several factors: (1) poverty of the chosen preference model (i.e., it does not include relevant features); (2) inadequacies in the elicitation; (3) lack of users' awareness of their preferences; and (4) inconsistencies in user scheduling decisions. Based on user feedback, model expressiveness is only found lacking in situations such as Request 3, exceptional cases for which the model was not designed. Based on the difference between more and less experienced subjects, (2) and (3) have significance. Based on the small deviations from subjects who undertook the exercise twice, (4) is found to have a small impact. Together, (2), (3), and (4) point to the benefit of revising the initial model by learning, supporting our overall paradigm. Lastly, although our lightweight model-based elicitation has advantages, the better performance of the default than elicited model instantiation for inexperienced users suggests that example-based elicitation may be valued.

6.3 Learning Experiments on Static User Data

The above study provided an opportunity to evaluate learning on real user data. We performed a leave-one-out cross-validation experiment on this data to see whether we could measure the effects of learning. We used each request in turn as the test example after training on the remaining requests, and then averaged the results over all the requests. To evaluate the effects of the elicited preferences on learning performance, we compared performance with the default and elicited model initialization. For the baseline (control) conditions, the training step was omitted, thus measuring the performance of the initial preferences only.

Unfortunately, the amount of data available in the user study was not sufficient to illustrate the effects of our learning algorithm.

What amount of data would be expected to determine whether online learning has an effect (positive or negative)? Figure 5 shows that scheduling with only five diverse candidates requires well over 20 instances to begin showing significant improvement with respect to the Spearman metric. Moreover, in real-world scenarios many of the candidates are very similar to or dominated by others. Thus in Figure 5 even five candidates, being generated randomly, are equivalent to many more candidates in a real-world setting; hence the number of instances required for online learning to be exhibited is significantly larger than 20. In contrast, the study of Section 6.2 made available only 11 instances.

6.4 Evaluation of Constraint Reasoner Performance

One requirement for our model was that it be simple enough to allow the PTIME Constraint Reasoner to find the top N solutions ($N = 35$ in our case) in a reasonable amount of time for an interactive system. For scheduling problems involving four participants, four meetings per user, and a two-day time period, the average time to find the top 35 solutions is 0.4 seconds.[6] Increasing the number of participants to six and then ten, the average times are 4.3 seconds and 14.4 seconds, respectively. A typical six-participant problem has approximately 225 constraints for each meeting, 140 of which are either disjunctive or contained preferences (non-disjunctive constraints without preferences do not need to be reasoned about during search, and therefore have negligible impact on reasoning time).

As the problem size increases, the time to find the optimal solution increases dramatically. However, our MC-DTPP algorithm still finds high-quality solutions for the largest problems we encounter (10 users, 20 meetings per user in the relevant time range). We ensure a responsive user experience by imposing a time limit on the reasoning, returning as the candidate set the best of the scheduling options found in that time. The computation is anytime: the system continues to search for further options as the user views those already found, and the user can explicitly request the system to “Search for More” options.

6.5 Learning Experiments on Dynamic User Data

Our final study evaluated data collected from two separate scheduling sessions. The intent was twofold. First, to investigate particularly the third and fifth questions posed earlier: learning performance using feedback from real users on interesting scheduling instances, and overall system performance in real-world situations. Second, to attempt to collect larger data sets than in our earlier study with static user data. As we will explain, we were not successful in this second aim, and thus the first aim could only be partially addressed.

The first source of data for this endeavor was a pilot study with four users, lasting for two hours. Each subject was asked to perform the preference elicitation step and then asked to schedule as many meetings as possible within the remaining time. Prior to the study, the calendar of each user was populated with a small number of meetings in order to increase the likelihood of non-trivial scheduling situations.

The subjects were asked to schedule the meetings as they would with their own calendars. Specifically, they were to issue meeting requests to the system like they would in real life, and to choose the schedules that they normally would. This resulted in two main differences from

[6] The experiments were run on a 1.7GHz Pentium M with 2GB RAM.

the previous user study: (1) the scheduling scenarios were closer to the individuals' real-life situations, and (2) not all the scheduling instances were "interesting" in the sense of exploring a wide variety of trade-offs between criteria. For example, if a user enters the request *"meeting re: 'project status' next Wednesday with Ken Doran"* and both the user and Ken are free most of next Wednesday, several meeting times are likely to be equally acceptable to the user. Not only will the learned model already be likely to rank an acceptable meeting time highly (negative examples provide much more information for learning), but even a poor model would have a good chance of ranking an acceptable meeting time highly as well.

The fact that not all scheduling situations are "interesting" has mixed implications for the user experience. Since our Constraint Reasoner will tend to produce preferred solutions for the easy instances regardless of the quality of the learned model, the user will get the impression that the system understands their preferences, and is thus likely increase their trust in it. However, since the learner gets informative data only a fraction of the time, the system will be slower to converge on the true (and possibly changing) preferences. In this short user study, given an average of ten scheduling requests made by each participant, only two or three proved useful to our learner. Furthermore, although up to 35 solutions are presented for each scheduling request, the average predicted rank of the user's choice was 2.54; that is, an acceptable schedule was usually in the first three presented. Thus, there was very little room for learning to improve performance. Indeed, a leave-one-out cross-validation experiment run on the data collected in this study showed no clear performance improvement with learning. The early rank of the subjects' choices means that the subjects may have simply selected the first acceptable option, rather than the "best" one. This claim is supported by [32], which claims that scheduling is often more about "satisficing" instead of "optimizing".

The data for the second part of our study with dynamic user data came from a week-long formal evaluation of the CALO system. Sixteen subjects participated in the evaluation. Each worked with a CALO agent under realistic conditions for a Critical Learning Period (CLP) that lasted over a week. Questions were administered to two flavours of each CALO after the end of the CLP. The first flavour, LCALO, made use of the knowledge it learned during the CLP; the second, BCALO, was stripped of such knowledge.

A set of parameterized questions were instantiated for and asked of each LCALO and BCALO pair. The scores were aggregated over a large set of such questions that were used to cover a broad range of cognitive assistant functions. Scoring was derived by comparing CALO's answers to the instantiated questions with answers elicited from the subjects in an explicit post data collection phase. The experimental design, the questions and their instantiations, and the scoring were conducted by an independent party.

Because the goal of the formal evaluation was to assess the learning ability of the overall CALO system, the subjects were not exercising PTIME continuously (in contrast to the subjects in the our own pilot studied described above). Despite the extended time frame of the data collection period in the experiment, the number of scheduling instances was not substantial. After accounting for technical difficulties with the system, the data of 13 subjects was usable. The number of instances per subject varied between 4 and 31; the mean number of instances was 13 and the median was 11.

We compared the answers given by BCALO, LCALO, and the users, to the three instantiations of one parameterized question. Each instantiated question asked for a total ranking of five candidate options to a given meeting request. BCALO uses the default, uninformative instantiation of the Choquet model. We also compared RCALO (random model instantiation) and ECALO (instantiation derived from the user-elicited preference statements).

Overall, the results on this data agree with the results from the data of our earlier smaller-scale study of Section 6.2. Model instantiation from elicitation is superior (by Spearman correlation of the rankings, in response to the question) to a random model instantiation; default and elicited model instantiations are similar. Online learning is observed, and might even help on more occasions than it hinders. Throughout, the deltas are small; at least to a large part this can be attributed again to the modest amount of data.

The true test of our balance between elicitation, learning, and constraint reasoning is how well our system performs on the complicated scheduling requests similar to those given in our paper-based study — the difficult over-constrained situations for which PTIME is designed. As the environment supporting our system becomes more stable, we will begin to collect data over a longer term and evaluate how quickly our system learns to rank schedules in the difficult cases. This limited study did reveal that, as a whole, the overall PTIME system provides a positive user experience even in cases where the learning has no effect.

7 RELATED WORK

In this section we concentrate on related work in automated and semi-automated calendaring systems. While both commercial (e.g., Outlook/Outlook Exchange Server, Sun Calendar Manager/Calendar Server) and open source (e.g., Zimbra, UWCalendar) calendaring systems abound to support centralized solutions within an institution, they leave the task of choosing a meeting time up to the user, thus avoiding the need to actively reason about user preferences. As an advanced example, *Meeting Maker* [36] supports the user in selecting a time by graphically depicting the availability of participants. The strength of this and other group calendaring systems is integration with institutional workflow (e.g., centralized calendar servers, room bookings); it is a tool rather than a scheduling assistant like PTIME.

While prior academic research has looked at one or more of the three aspects of modelling and eliciting, learning, and reasoning, the resulting systems have rarely sought to encompass all three. At the same time, researchers have considered the distributed meeting negotiation task, which we do not address in this report, from both theoretical ([4, 38]) and system ([39, 8, 3]) angles.

A system we will call *Tulsa* [13] implements earlier distributed constraint-based negotiation algorithms [39] with an emphasis on making the meeting scheduling process more efficient. While Tulsa accounts for a wide variety of user preferences, the preferences are not learned. Moreover, it remains to be proved that users are willing to view and manipulate a screen of scrollbars to provide the parameters for Tulsa's weighted sum preference model.

The value of applying machine learning to user scheduling preferences was recognized by Kozierok and Maes [20], presenting the *Learning Interface agent* (LIA). The system learns rules for how to respond to scheduling situations (e.g., accepting a meeting request or not), and preferred meeting times. The preference model for the latter is based on user day/time preferences, and user assessment of the importance of the preferences of other users. The suggested times are found not by constraint reasoning but by finding the timeslot with the maximum weighted preference score; non-temporal relaxations are not considered. In further contrast to our work, LIA requires explicit user feedback to correct mistakes. Its designers [20] emphasize the centrality of user confidence in an assistive agent, and thus the centrality of both adjustable autonomy and explanation, e.g., giving confidence levels for agent predictions.

Calendar Apprentice (CAP) [25] takes the antipodal approach. CAP offers no automated reasoning but provides scheduling assistance in the form of learned recommended values. The meeting request is elicited by a sequence of prompts (equivalent to a fixed-order filling in of a form); CAP offers a suggested value at each step and acquires training examples when the user overrides these values. Thus, the learning is unobtrusive, like PTIME, but in contrast to our work the learning is performed offline, and rules rather than a preference model are learned.

The same preference model necessary for meeting scheduling is of use in the wider personalized time management context. *Augur* [40], for instance, is a system designed to blend the flexibility of an individual's calendaring process while opening up interpersonal communication via shared calendars. Augur learns models of user event attendance, and augments a calendar display with iconic, colour, and transparency visualizations.

A similar group facilitation approach is found in *groupTime* [3]. groupTime models user temporal preferences and, as Augur, learns models of user attendance. In addition, groupTime offers suggested times for a meeting request, based on the participants' preferences and predicted attendance. The meeting request time is elicited by painting preferences onto a calendar. In contrast to PTIME, groupTime offers a web-based multi-phase negotiation process among the meeting invitees, in a university setting where there is equality among the participants (i.e., no designated host arranging the meeting). groupTime's reasoning is limited to visualizing the aggregated preference of each timeslot; suggested schedule options created by relaxing the request are not provided.

Like LIA, CAL, and groupTime, *RhaiCAL* [5] (a descendant of the RCAL scheduling agent [34]) presents alternate scheduling options on a calendar view. Like PTIME, RhaiCAL's mode of operation is to support a single meeting organizer, rather than the many-equal-participants use case of groupTime; like PTIME it is intended for the difficult case of over-constrained, multiple-person meetings. RhaiCAL's display of schedules is designed to handle over-constrained situations by indicating to the meeting host each participant's preferences. While relaxations of the meeting request are computed, these principally revolve around temporal constraints, in contrast with our sophisticated trade-offs. RhaiCAL employs *availability bars* [6], a interaction and visualization tool for temporal preferences. Such direct manipulation interface tools could complement our NL-based elicitation of meeting request constraints and preference. They could also inform the user as she selects among candidate schedules; our current visualization of scheduling options is based on similar principles to availability bars, but is invoked via an "Explain" option, rather than being integrated with a simultaneous calendar view of the candidate schedules.

A sibling of RhaiCAL, *CMRadar* [26] is the work most similar to the approach taken by PTIME. A CMRadar agent helps its user schedule meetings by parsing out meeting information from email. It generates schedule options using a constraint-based scheduler that takes into account user preferences and presents candidate schedules with a graphical visualization. CMRadar has a simple natural language parser based on templates that are used to interpret meeting-related emails. This mirrors our NL input mechanism, but is less flexible and on a smaller scale. The CMRadar preference model is a weighted sum over scheduling features, which is a restricted case of our Choquet form. CMRadar uses a passive learning approach to learn user preferences [30], which contrasts with PTIME's online, active learning.

8 CONCLUSION

Personalization is a key requirement for adoption of automated scheduling technology. A model of the user's preferences, on one hand, must be expressive enough to capture the salient features that distinguish one scheduling option from another in the user's judgment. On the other hand, the model must be amenable to elicitation to populate instances of it, to explanation, and to reasoning over scheduling requests to derive candidate schedules. Moreover, if the system is to adapt itself, the model must further be amenable to the machine learning techniques employed.

Based on an initial user study, we devised a preference model designed to balance these needs. We have implemented this model, which is based on a Choquet integral form, in a deployed scheduling assistant agent. The implementation necessitated a user interface design that exposes the richness of the model without overbearing the user, novel constraint-based reasoning to find optimal schedules according to the model, and adaption of earlier work in non-obtrusive online learning to update the model as the user employs the system. A distinguishing feature of the model developed is the ability to express sophisticated trade-off between schedule features; capturing such trade-offs is crucial if a system is to offer the user desirable relaxations of over-constrained meeting requests, the very situation where scheduling assistance is potentially the most valuable.

The inconclusive results of our experiments point to the difficulty of evaluating a model and learning process. Positively, the user interviews indicate that our model is expressive enough to capture user scheduling preferences, to a degree sufficient for the types of meeting requests made by knowledge workers in a typical office setting. Although we find that learning obtains an inexact representation of the schedule ranking function (as measured by Spearman's ρ), the refined preference model obtained by the learning becomes adept at suggesting the most preferred schedules.

Users found that the implemented system provides reasonable scheduling options from its first use and exhibits increasing trustworthiness over time — complementary aspects found essential if scheduling technology is to be adopted in practice [25]. Integration with the user's existing calendaring systems and workflow combine with the low demand of the preference elicitation and learning, to further support adoption of the system [32, 2]. An extensive formal study with real users in a deployed setting indicates user satisfaction with the overall system.

These positive points said, the experiments were not able to evaluate the success of the our paradigm of lightweight elicitation of an initial model, and its subsequent non-intrusive refinement. While users found the system's behaviour satisfactory, the evaluation with human subjects did not demonstrate that an elicited instantiation of the preference model is superior to a random instantiation, nor that online learning converges to the user's true model in practical settings (although in artificial settings it does). In other words, the experiments were unable to determine the correlation or otherwise between the seen satisfactory performance of the system and the model, elicitation, and learning approach underlying it.

The reasons for this inability, as noted earlier, include the small number of subjects, a lack of data for learning to be visible, the difference between "interesting" and "uninteresting" problem instances, and the difficulty of uncovering the user's true preferences or capturing them by approximate metrics.

A central aspect, therefore, of our ongoing work is to design experiments that are able to answer the above questions. Toward this goal, we have begun designing a test harness that will allow us to quickly gather new data and experiment with various preference models and learning algorithms. The harness will enable more realistic validation before deployment and will help characterize how well different model and algorithm combinations work for different types of scheduling instances.

In ongoing work we are considering the use of multiple complementary preference models. For instance, deploying a simple 1-order Choquet model that is more readily elicited and refined, for under-constrained problem instances, in conjunction with the existing 2-order model that is able to capture criteria trade-offs, for critically- and over-constrained instances where the interaction between criteria is a much more significant factor.

On the offline learning side, we are investigating the merits of more direct, albeit heavyweight, elicitation of the Choquet complementarity and substitutability degrees between criteria, via methods from decision theory. On the online learning side, we are investigating hierarchical learning over the criteria within the Choquet u_i functions as well as over the aggregated F function. We are also exploring adaptive presentation of the candidate set of schedule options, including learning what options to present (e.g., varying the diversity of the set according to measures of its informational entropy [42]).

Other avenues of interest include learning the importance of a meeting to an individual and of an individual to a meeting (although not described in this report, PTIME has an initial capability in both areas), predicting the likelihood of a participant's attendance at a meeting (as for instance done by [40]), facilitating negotiation with other users to refine or modify the organizer's chosen meeting option, exploring visualization of others' preferences and predicted attendance ([3, 6]), and transfer learning of typical preference models between users ([3]). Last, we are working on explaining the system's conclusions based on the Choquet model, in the context of expanding our work from meeting scheduling to wider calendar management and negotiation processes.

Acknowledgments

The authors thank their colleagues on the CALO project, at SRI and elsewhere. This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. NBCHD030010. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the Defense Advanced Research Projects Agency (DARPA), or the Department of Interior-National Business Center (DOI-NBC).

References

[1] BEARD, D., PALANIAPPAN, M., HUMM, A., BANKS, D., NAIR, A., AND SHAN, Y.-P. A visual calendar for scheduling group meetings. In *Proc. of ACM Conf. on Computer-supported Cooperative Work (CSCW'90)* (1990), pp. 279–290.

[2] BERRY, P., CONLEY, K., GERVASIO, M., PEINTNER, B., URIBE, T., AND YORKE-SMITH, N. Deploying a personalized time management agent. In *Proc. of AAMAS'06 Industrial Track* (2006), pp. 1564–1571.

[3] BRZOZOWSKI, M., CARATTINI, K., KLEMMER, S. R., MIHELICH, P., HU, J., AND NG, A. Y. groupTime: preference-based group scheduling. In *Proc. of CHI-06* (2006), pp. 1047–1056.

[4] EPHRATI, E., ZLOTKIN, G., AND ROSENSCHEIN, J. A non-manipulable meeting scheduling system. In *Proc. of Thirteenth Intl. Distributed Artificial Intelligence Workshop* (1994).

[5] FAULRING, A., AND MYERS, B. A. Enabling rich human-agent interaction for a calendar scheduling agent. In *Proc. of CHI-05* (2005), pp. 1367–1370.

[6] FAULRING, A., AND MYERS, B. A. Availability bars for calendar scheduling. In *Proc. of CHI-06* (2006), pp. 760–765.

[7] FISHBURN, P. Interdependence and additivity in multivariate, unidimensional expected utility theory. *Intl. Economic Review 8* (1967), 335–342.

[8] GARRIDO, L., BRENA, R., AND SYCARA, K. Multiple negotiation among agents for a distributed meeting scheduler. In *Proc. of Fourth Intl. Conf. on Multi-Agent Systems* (2000), pp. 55–72.

[9] GERVASIO, M. T., MOFFITT, M. D., POLLACK, M. E., TAYLOR, J. M., AND URIBE, T. E. Active preference learning for personalized calendar scheduling assistance. In *Proc. of IUI'05* (2005), pp. 90–97.

[10] GOEBEL, R., GREINER, R., AND LIN, D., Eds. *Computational Intelligence: Special Issue on Preferences*, vol. 20. 2004.

[11] GOOGLE, INC. Google calendar `calendar.google.com`, 2006.

[12] GRABISCH, M., MUROFUSHI, T., AND SUGENO, M., Eds. *Fuzzy Measures and Integrals — Theory and Applications*. Physica-Verlag, Heidelberg, 2000.

[13] HAYNES, T., SEN, S., ARORA, N., AND NADELLA, R. An automated meeting scheduling system that utilizes user preferences. In *Proc. of First Intl. Conf. on Autonomous Agents* (1997), pp. 308–315.

[14] HIGA, K., SHIN, B., AND SIVAKUMAR, V. Meeting scheduling: an experimental investigation. In *Proc. of IEEE Intl. Conf. on Systems, Man, and Cybernetics* (1996), pp. 2023–2028.

[15] HOGG, R. V., AND CRAIG, A. T. *Introduction to Mathematical Statistics*. Macmillan, New York, 1995.

[16] HORVITZ, E. Principles of mixed-initiative user interfaces. In *Proc. of CHI-99* (1999), pp. 159–166.

[17] JOACHIMS, T. Optimizing search engines using clickthrough data. In *Proc. of ACM Conf. on Knowledge Discovery and Data Mining (KDD'02)* (2002).

[18] JUNKER, E. QuickXplain: Preferred explanations and relaxations for over-constrained problems. In *Proc. of AAAI-04* (2004), pp. 167–172.

[19] KEENEY, R. L., AND RAIFFA, H. *Decisions with Multiple Objectives*. Cambridge University Press, Cambridge, 1993.

[20] KOZIEROK, R., AND MAES, P. A learning interface agent for scheduling meetings. In *Proc. of IUI'93* (1993), pp. 81–88.

[21] LABREUCHE, C., AND GRABISCH, M. The Choquet integral for the aggregation of interval scales in multicriteria decision making. *Fuzzy Sets and Systems 137*, 1 (2003), 11–26.

[22] LABREUCHE, C., AND HUÉDÉ, F. L. MYRIAD: a tool for preference modeling application to multi-objective optimization. In *Proc. of CP'05 Workshop on Preferences and Soft Constraints (Soft'05)* (2005).

[23] LABREUCHE, C., AND HUEDE, F. L. Inconsistencies in the determination of a capacity. In *Proc. of ECAI'06 Multidisciplinary Workshop on Advances in Preference Handling* (2006).

[24] MARICHAL, J.-L., AND ROUBENS, M. Determination of weights of interacting criteria from a reference set. *European J. of Operational Research 124*, 3 (2000), 641–650.

[25] MITCHELL, T., CARUANA, R., FREITAG, D., MCDERMOTT, J., AND ZABOWSKI, D. Experience with a learning personal assistant. *Comm. of ACM 37*, 7 (1994), 80–91.

[26] MODI, P. J., VELOSO, M. M., SMITH, S. F., AND OH, J. CMRadar: A personal assistant agent for calendar management. In *Proc. of Agent-Oriented Information Systems (AOIS'04)* (2004), pp. 169–181.

[27] MOFFITT, M. D., PEINTNER, B., AND YORKE-SMITH, N. Multi-criteria optimization of temporal preferences. In *Proc. of CP'06 Workshop on Preferences and Soft Constraints (Soft'06)* (2006).

[28] MOFFITT, M. D., AND POLLACK, M. E. Temporal preference optimization as weighted constraint satisfaction. In *Proc. of AAAI'06* (2006).

[29] MYERS, K., BERRY, P., BLYTHE, J., CONLEY, K., GERVASIO, M., MCGUINNESS, D., MORLEY, D., PFEFFER, A., POLLACK, M., AND TAMBE, M. An intelligent personal assistant for task and time management. *AI Magazine 28* (2007). To appear.

[30] OH, J., AND SMITH, S. F. Learning calendar scheduling preferences in hierarchical organizations. In *CP'04 Workshop on Preferences and Soft Constraints (Soft'04)* (2004), pp. 147–161.

[31] OZTÜRK, M., TSOUKIÀS, A., AND VINCKE, P. Preference modelling. In *State of Art in Multiple Criteria Decision Analysis*, M. Ehrgott, J. Figuiera, and X. Gandibleux, Eds. Springer-Verlag, 2005, pp. 27–72.

[32] PALEN, L. Social, individual and technological issues for groupware calendar systems. In *Proc. of CHI-99* (1999), pp. 17–24.

[33] PAYNE, J., BETTMAN, J., AND JOHNSON, E. *The Adaptive Decision Maker*. Cambridge University Press, Cambridge, 1993.

[34] PAYNE, R., SINGH, R., AND SYCARA, K. Rcal: A case study on semantic web agents. In *Proc. of AAMAS'02* (2002), pp. 802–803.

[35] PEINTNER, B., AND POLLACK, M. E. Low-cost addition of preferences to DTPs and TCSPs. In *Proc. of AAAI'04* (2004), pp. 723–728.

[36] PEOPLECUBE, INC. Meeting Maker http://www.peoplecube.com/products/meetingmaker/, 2007.

[37] RICH, C., SIDNER, C., LESH, N., GARLAND, A., BOOTH, S., AND CHIMANI, M. DiamondHelp: A collaborative interface framework for networked home appliances. In *Proc. of IEEE Intl. Conf. on Distributed Computing Systems Workshops* (2005), pp. 514–519.

[38] SANDIP, S., AND DURFEE, E. A formal study of distributed meeting scheduling. *J. Group Decision and Negotiation 7* (1998), 265–298.

[39] SEN, S., HAYES, T., AND ARORA, N. Satisfying user preferences while negotiating meetings. *Intl. J. of Human Computer Studies 47* (1997), 407–427.

[40] TULLIO, J., GOECKS, J., MYNATT, E., AND NGUYEN, D. Augmenting shared personal calendars. In *Proc. of ACM Symposium on User Interface Software and Technology (UIST'02)* (2002), pp. 11–20.

[41] VIAPPIANI, P., FALTINGS, B., AND PU, P. Evaluating preference-based search tools: a tale of two approaches. In *Proc. of AAAI-06* (2006), pp. 205–211.

[42] WEBER, J. S., AND POLLACK, M. E. Entropy-driven online active learning for interactive calendar management. In *Proc. of IUI'07* (2007), pp. 141–150.